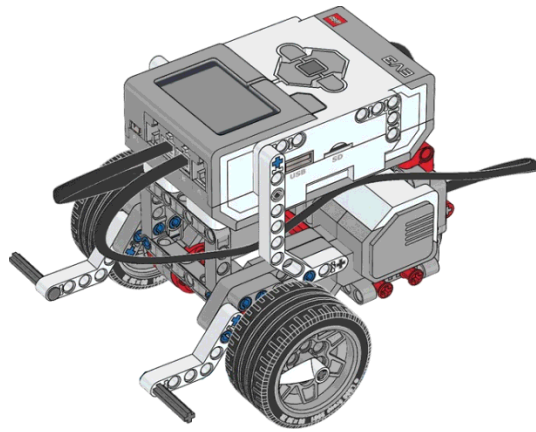


# LUNAR ROVER

## SOFTWARE DESIGN DOCUMENT

Software Engineering & Project



**Team: PG-29**

Benjamin Winding

Kin Leong Lee

Pavitterjeet Sidhu

Phan Huy Nguyen

Sean Hennessy

Xiaoshan Chen

3/10/2017

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Purpose . . . . .	4
1.2	Scope . . . . .	4
1.3	References . . . . .	4
1.4	Overview . . . . .	4
1.5	Constraints . . . . .	4
<b>2</b>	<b>System Overview</b>	<b>6</b>
2.1	Robot . . . . .	6
2.2	Computer . . . . .	6
2.2.1	GUI . . . . .	6
2.2.2	Manual Controller . . . . .	6
2.2.3	Auto Controller . . . . .	6
2.2.4	Map . . . . .	6
2.3	Integration . . . . .	7
<b>3</b>	<b>System Architecture &amp; Component Design</b>	<b>8</b>
3.1	Software System Overview . . . . .	8
3.2	Component Decomposition Description . . . . .	9
3.3	Detailed Components Design Description . . . . .	10
3.3.1	C00 Lejos Library . . . . .	10
3.3.2	C01 Movement Service . . . . .	10
3.3.3	C02 Sensors Service . . . . .	11
3.3.4	C03 Robot State Machine . . . . .	11
3.3.5	C04 Shared Core . . . . .	11
3.3.6	C05 User Interface . . . . .	11
3.3.7	C06 UI Updater . . . . .	12
3.4	Architectural Alternatives . . . . .	12
3.5	Design Rationale . . . . .	12
3.5.1	Initial Testing . . . . .	12
3.5.2	Component Development . . . . .	13
3.5.3	Dependency Minimisation . . . . .	13
<b>4</b>	<b>Data Design</b>	<b>14</b>
4.1	Database Description . . . . .	14
4.2	Data Structures . . . . .	14
<b>5</b>	<b>Design Details</b>	<b>16</b>
5.1	Class Diagram . . . . .	16
5.2	State Diagram . . . . .	17
5.3	Interaction Diagram . . . . .	17
<b>6</b>	<b>Human Interface Design</b>	<b>19</b>
6.1	Overview of the User Interface . . . . .	19
6.2	Detailed Design of the User Interface . . . . .	19
6.2.1	GUT's screen image . . . . .	19
6.2.2	Manual Control . . . . .	20

6.2.3	Map panel . . . . .	20
6.2.4	Other Functionalities . . . . .	23
<b>7</b>	<b>Resources Estimate</b>	<b>25</b>
7.1	Budget . . . . .	25
7.2	Manpower . . . . .	25
7.3	Software . . . . .	25
7.4	Hardware . . . . .	25
7.5	Facilities . . . . .	25
<b>8</b>	<b>Definitions, Acronyms and Abbreviations</b>	<b>26</b>
8.1	Definitions . . . . .	26
8.2	Acronyms . . . . .	26
8.3	Abbreviations . . . . .	26

## List of Figures

1	Detailed operating system requirements . . . . .	5
2	Physical System Overview . . . . .	6
3	Software Architectural Overview - V1 . . . . .	8
4	Software Architectural Overview - V1 . . . . .	9
5	Basic Program Flow Overview . . . . .	9
6	Database ER Model . . . . .	15
7	Class diagram of the whole system . . . . .	16
8	The State Diagram of the whole system . . . . .	17
9	The interaction diagram of the whole system . . . . .	18
10	Graphic User Interface . . . . .	20
11	Manual control . . . . .	21
12	Map Panel Components . . . . .	22
13	Status Panels . . . . .	22
14	Other Menu Functions . . . . .	23
15	Help Popup Window . . . . .	23
16	About Popup Window . . . . .	24

## List of Tables

1	High Level Component Overview . . . . .	9
2	The required threads determined through testing . . . . .	13
3	GUI Features . . . . .	19
4	Method called for Manual Control . . . . .	21
5	Methods called by map listeners . . . . .	22

## Revision History

Name	Date	Version	Summary of Changes
Ben	2-Oct-2017	0.10	Initial Draft
Huy Nguyen	3-Oct-2017	0.12	Add section 5 Design Detail draft
Ben	2-Oct-2017	0.13	Revised the figures position
Huy Nguyen	10-Oct-2017	0.14	Revise section 5 Design Detail - Add context for diagrams
Issac Lee	3-Oct-2017	0.15	Fixed some symbols display issue and Revised the introduction and resource estimate sections
Issac Lee	25-Oct-2017	0.16	Update GUI design
Ben	30-Oct-2017	0.2	Added content to Component Breakdown
Issac Lee	30-Oct-2017	1.0	Release Final version

# 1 Introduction

An overview of the SDD and the scope of the system will be discussed in this section.

## 1.1 Purpose

The purpose of this document is to analysis the Robot Mapping System(RMS) that is defined in the SRS document. The RMS is for a prototype rover capable of surveying a designated area automatically, and developing an appropriately safe system for the client, and it shall be constructed in real time as long as the robot is surveying, showing lines, obstacles and current location of the robot. The details of RMS will be discussed in the following, mainly including the system architecture and components design, data design, and human interface design.

## 1.2 Scope

The RMS, real-time system, involves building a prototype rover capable of surveying a designated area automatically, which is controlled via a remote location. The map can store the map information, including the current location of the robot. Also, NGZ, obstacles with colour lines and the boundaries of the map will be shown on the map. The RMS will collect map information, such as detecting obstacles or NGZ, showing on the map with different colours. The particular map can be zoomed and recovery the map after zooming in and zooming out. The map can be exported with XML file. In this project, the prototype rover is based on A1 size paper.

## 1.3 References

- Software Requirements Specification.
- Software Project Management Plan.
- SDD template.

## 1.4 Overview

The description of RMS is included in the SDD document. The system architecture and its components and data design will also be delivered in this document. In addition, some class diagrams, state diagrams, and interactions diagrams will be included in the document. However, the implementation of the programming, the explanation of algorithms and the consequences of testing are not included in SDD document.

## 1.5 Constraints

### Hardware Constraints

- The software can be run on Windows, Mac or Ubuntu systems, specific details are shown in fig 1.
- WiFi access is required to run the control software.
- The GUI must be compiled using JDK (version 1.8 only).

OS	Minimum Version	Requirements
Windows	7	1 (GHz) or faster 32-bit (x86) or 64-bit (x64) processor*; 1 gigabyte (GB) RAM (32-bit) or 2 GB RAM (64-bit); 16 GB available hard disk space (32-bit) or 20 GB (64-bit); DirectX 9 graphics device with WDDM 1.0 or higher driver.
Mac	OS X 10	An Intel Core 2 Duo, Core i3, Core i5, Core i7, or Xeon processor; 7 GB of available disk space; 2 GB of RAM.
Ubuntu	16.10	1 (GHz) or faster 32-bit (x86) or 64-bit (x64) processor*; 1 gigabyte (GB) RAM (32-bit) or 2 GB RAM (64-bit); 16 GB available hard disk space (32-bit) or 20 GB (64-bit);

Figure 1: Detailed operating system requirements

#### Software Constraints

- The embedded software on the rover is to be written in Java, using the LeJOS Ev3 library version:0.9.0-beta.
- The build tool for compiling the software and deploying it to the system is *ant*.
- The university enterprise Github instance is the version control system to be used throughout the project.
- During development the project team will use the IntelliJ IDE to design and write code.
- Latex is used to produce documentation.
- Accuracy of sensors, such as the colour sensor and ultrasonic sensor.

## 2 System Overview

In this project, we are going to develop a software which will provide the requested behaviours in the Lego Mindstorms EV3 robot. The system overview is shown in figure 2, and is briefly described in the following subsections.

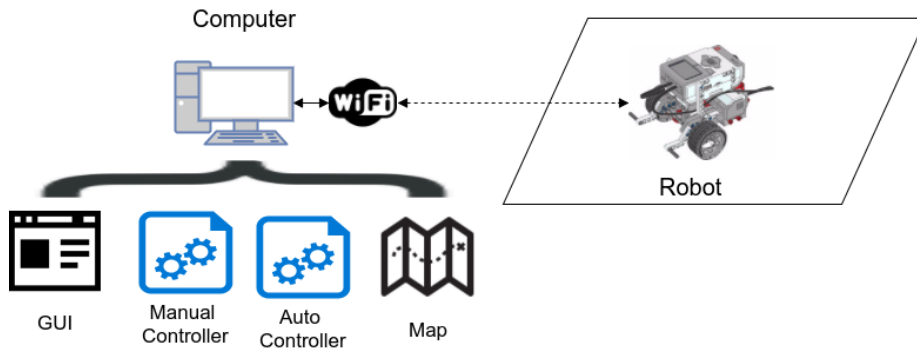


Figure 2: Physical System Overview

### 2.1 Robot

A robot will be assembled from the parts provided by the client which should be able to performed all the activities defined in the user requirement section of SRS document.

### 2.2 Computer

The computer is a platform for several other parts of the system. Including the following sections.

#### 2.2.1 GUI

Graphical user interface will be developed to enable the user to interact with system. Buttons and map scene will be provide to control and configure the robot.

#### 2.2.2 Manual Controller

Manual controller is another subsystem of the project which will enable the robot operator to control the robot manually by using the the provided GUI.

#### 2.2.3 Auto Controller

Auto controller will enable the robot to perform some operations like doing survey, avoid obstacles etc. automatically.

#### 2.2.4 Map

The survey map is one of main component of the system. The software will enable the user to load and save the survey map in XML format. The software will also enable the user to zoom on a particular area in the map.

## **2.3 Integration**

Integration will be the key behind the success of this system. Proper integration is required between the above mentioned system components. For connectivity between software and robot, the system will provide different means such as WiFi and bluetooth.



### 3 System Architecture & Component Design

#### 3.1 Software System Overview

The software architecture was developed through an iterative process. After testing a prototype of the system, it was clear that a multi-threaded design for the system would be ideal and allow different components to run simultaneously. Figure 3 shows the initial multi-threaded architecture, which was improved upon during development as shown in Figure 4. Finally an overview of the program flow of the system can be seen in Figure 5.

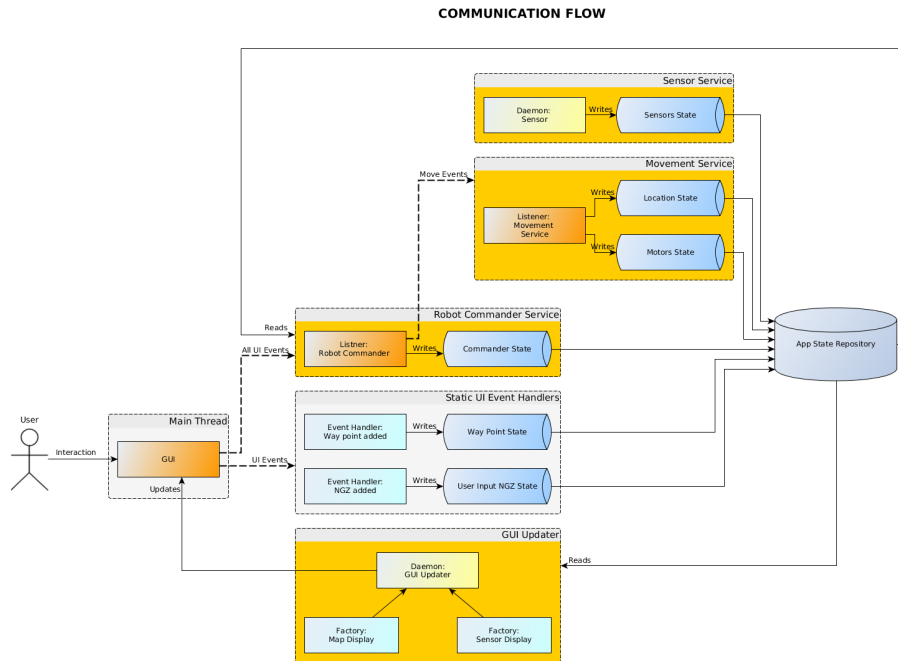


Figure 3: Software Architectural Overview - V1

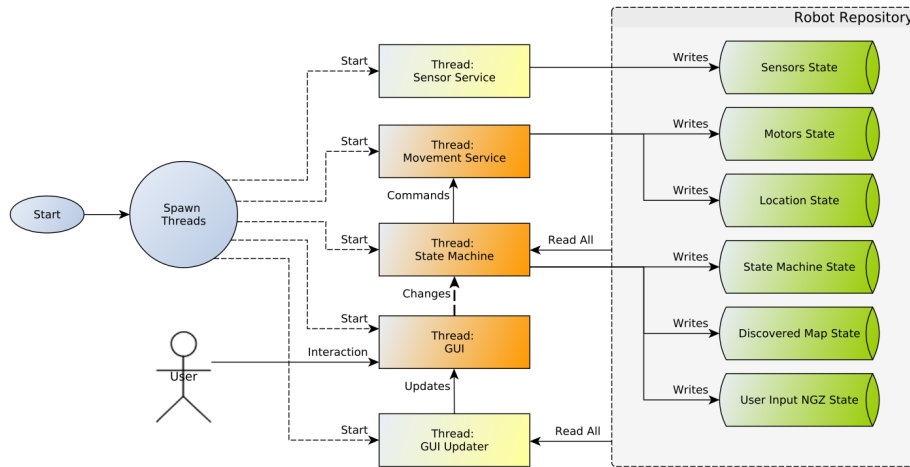


Figure 4: Software Architectural Overview - V1

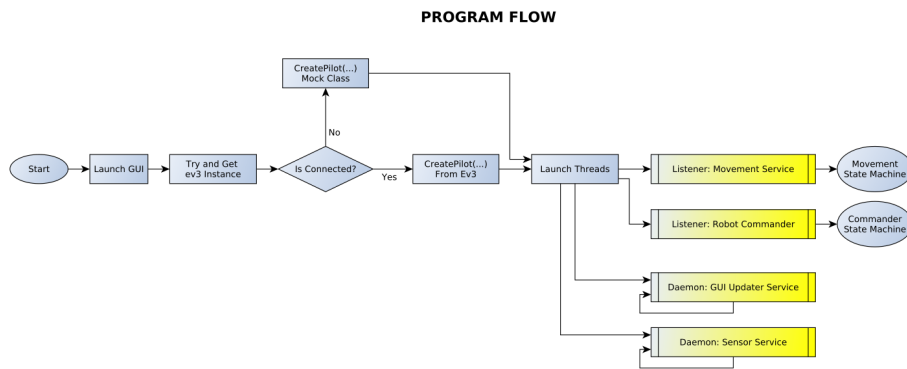


Figure 5: Basic Program Flow Overview

### 3.2 Component Decomposition Description

The system has been broken down into clearly defined components, which can be tested individually. Each of these components perform a function in the entire system.

Location	Description
Ev3 Robot	Embedded control software
PC	UI display
PC	Robot Control Services
PC	Robot State Logic

Table 1: High Level Component Overview

### **Embedded Control Software**

This component of the architecture is provided by the 3rd party library, version Lejos 0.9. The code will not be changed in this project.

### **UI Display**

The main interface for the user of the system is the UI Display. This part of the system is

### **Robot Control Services**

The robot control services represent a part of the architecture that interact with the robot directly. It interacts with the Lejos library which provides a simplified abstraction for other parts of the system. Services in this include:

- Movement Service
- Sensor Service

### **Robot State Logic**

The core of the robot's behaviour is maintained in a part of the system called the Robot State Logic. This provides a centralised location to describe the behaviour of the robot. It acts as an adapter between the UI and the Robot control services.

## **3.3 Detailed Components Design Description**

The following sections show the detailed design of each of the components in the system. It shows the Purpose, Function, Subordinates, Dependencies, Interfaces and Data that are required for the component.

### **3.3.1 C00 Lejos Library**

- Purpose: UR01,UR08,UR09,UR10,UR11,UR12,UR13
- Function: To interface to the robot.
- Subordinates: C01 Movement Service, C02 Sensors Service
- Dependencies: Null
- Interfaces:
  - In >> ArcMoveController
  - Out << Ev3Remote
  - Out << Pose
  - Out << RMISampleProvider
- Data:

### **3.3.2 C01 Movement Service**

- Purpose: UR01,UR12,UR13,UR15,UR02
- Function: To provide a simple abstraction to move the robot.
- Subordinates: C03 Robot State Machine
- Dependencies: C00 Lejos Library
- Interfaces:
  - Out: LocationState >> location data
  - In: LejosLibrary << lejos functions
  - In: MovementThread << Move commands
- Data: LocationState

### 3.3.3 C02 Sensors Service

- Purpose: UR08,UR09,UR10,UR11,UR02
- Function: To provide a simple abstraction to receive sensor readings from the robot.
- Subordinates: C03 Robot State Machine
- Dependencies: C00 Lejos Library, C04 Shared Core
- Interfaces:
  - Out: `SensorsState << sensor readings`
  - In: `LejosLibrary >> lejos functions`
- Data: `SensorsState`

### 3.3.4 C03 Robot State Machine

- Purpose: UR03,UR08,UR09,UR10,UR11
- Function: A location to define the logic what the robot does in the system.
- Subordinates: C05 User Interface
- Dependencies: C04 Shared Core, C01 Movement Service, C02 Sensors Service
- Interfaces:
  - In: `SensorsState >> input into statemachine`
  - In: `LocationState >> input into statemachine`
- Data: `Null`

### 3.3.5 C04 Shared Core

- Purpose: UR03,UR14
- Function: This is a component to share interfaces between other components, like models and service instances.
- Subordinates: C03 Robot State Machine, C05 User Interface, C02 Sensors Service, C06 UI Updater, C01 Movement Service
- Dependencies: `Null`
- Interfaces:
  - Out: `StateRepository >> output state for other parts of system`
  - Out: `ServiceManager >> output service instances for other parts of system`
- Data:
  - `MapPoint`: shared class to represent a point on the map
  - `Colour`: shared class to represent a colour on the map

### 3.3.6 C05 User Interface

- Purpose: UR03,UR04,UR07,UR05,UR06
- Function: To provide an interactive display of the system to the user.
- Subordinates: C06 UI Updater
- Dependencies: C03 Robot State Machine, C04 Shared Core
- Interfaces:
  - In: `User interaction << Mouse clicks, key presses, handled in UI code-behind.`
  - Out: `CommandEvents >> commands to be handled by Robot State Machine`

- Out: MapEvents >> commands to be handled by UIUpdater
- Data:
  - MapPoint: shared class to represent a point on the map
  - Colour: shared class to represent a colour on the map

### 3.3.7 C06 UI Updater

- Purpose: UR02,UR03,UR04,UR07,UR05,UR06
- Function: To constantly update parts of the GUI in real time.
- Subordinates:
- Dependencies: C05 User Interface, C03 Robot State Machine, C04 Shared Core
- Interfaces:
  - In: UiComponent >> Allows the UIUpdater to manipulate it
  - Out: UiComponent << After the UIUpdater has finished making the relevant updates
- Data:
  - MapPoint: shared class to represent a point on the map
  - Colour: shared class to represent a colour on the map

## 3.4 Architectural Alternatives

There are several other architectural alternatives to the one that was chosen for this system.

### Single Threaded Event Loop

A viable alternative is the single threaded event loop. In this architecture, the event loop would be continually running and functions would be called periodically based on events. This would be an efficient system, however it is important that every call is non-blocking which is hard to implement and maintain in practice, it's for this reason that a multi-threaded system was desired.

### Monolithic System

Another alternative is to combine the system code into large files. This actually reduces the amount of overall code in the system, however the code becomes hard to maintain as the code is highly interdependent. This is one of the main reasons we decided to break apart the system into multiple services.

## 3.5 Design Rationale

The system design emerged from iterative design and testing during the initial stages of the project. The rationale behind the system design can be explained by illustrating the different stages of the project.

### 3.5.1 Initial Testing

During the beginning of the system's design the architecture was very fragile, as different parts of the system were being explored in order to determine which components were necessary. It was found that several parts of the system would need to be running concurrently. The required threads are shown in Table 2.

<b>SensorsThread</b>	Fetching the sensor information
<b>MovementThread</b>	Moving the robot, tracking location
<b>LogicThread</b>	Handling state changes and movement subroutines
<b>GUI</b>	Interface for users to interact with the system

Table 2: The required threads determined through testing

### 3.5.2 Component Development

In this phase, the system was divided into logical components which had specific purposes. This allowed us to work on parts of the project independently in order to maximise project efficiency. The components that were defined also include the implementations for the threads determined in the initial testing stage. They are outlined in detail in section 3.3.

### 3.5.3 Dependency Minimisation

This was a crucial phase of system development, where the dependencies of each of the components were minimised and centralised, so that each of the components could be revised independently. This provided the system with clearly defined minimal components, which also provided easier abstractions for testing, as dependencies were minimised.

## **4 Data Design**

### **4.1 Database Description**

This section describes the data base which will be used by the Ev3 Robot Controller program. The database consists of a repository of XML files that hold the information on the graphical representation of the map as shown in the GUI. The repository will hold the XML files which are exported by the user. The location XML files which the user will be importing will vary and will not be stored by the program, it will simply be converted to an object that will replace the current map on the GUI.

### **4.2 Data Structures**

The data structure is represented by an ER diagram below. Note that the GUI map entity has no unique identifier. This is because the GUI map will be in constant state of change and only needs to be a snapshot on the export command.

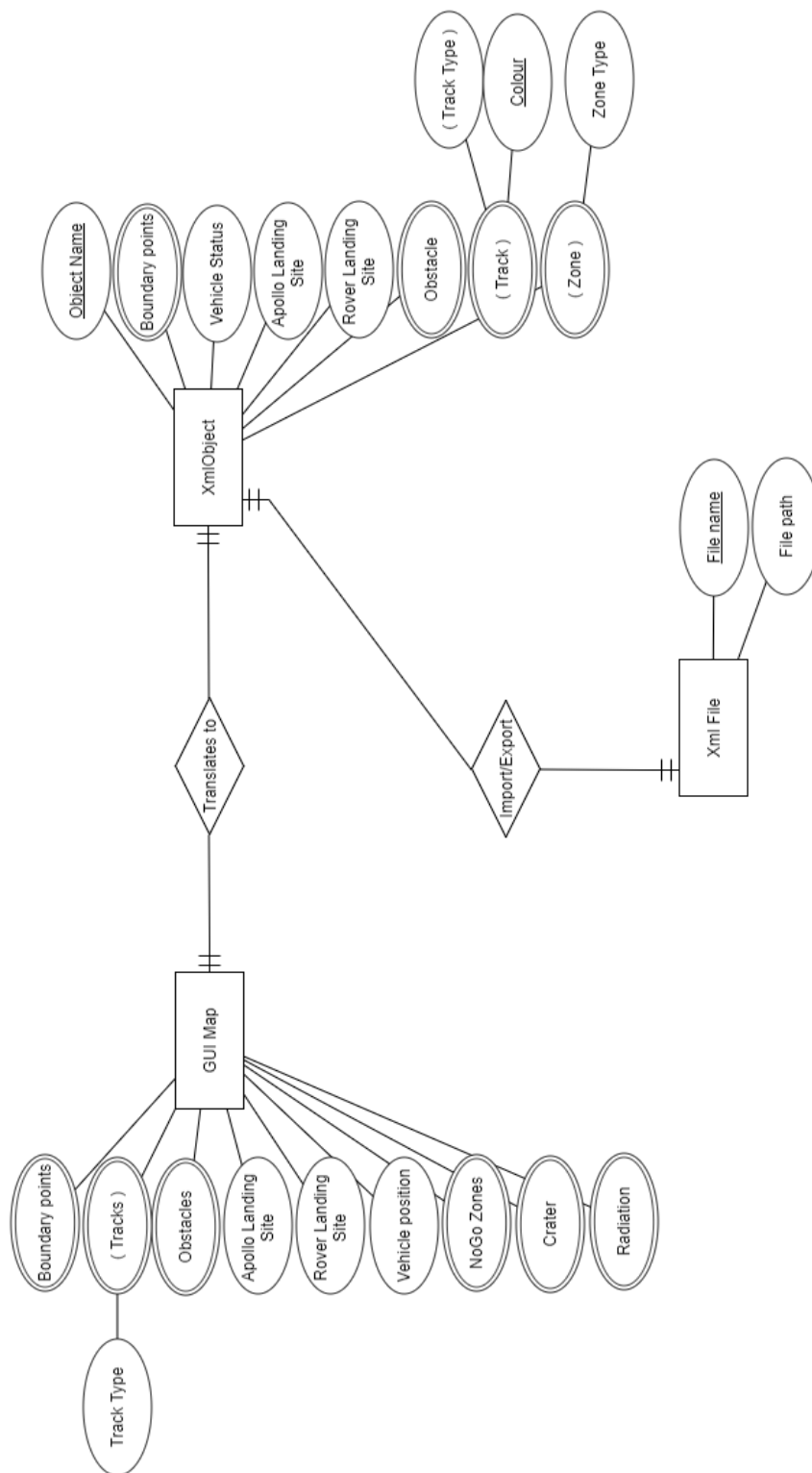


Figure 6: Database ER Model  
15



## 5 Design Details

### 5.1 Class Diagram

The picture below shows the class diagram of Lunar Rover project. There 4 main important classes in this project:

- AppStateRepository class hold all data imported/collected in application
- RobotConnectionService and RobotMovementService responsible for connecting and control movement of robot.
- ServiceManager responsible for managing services use in application
- StateMachineBuilder responsible for controlling and switching between states of robot.

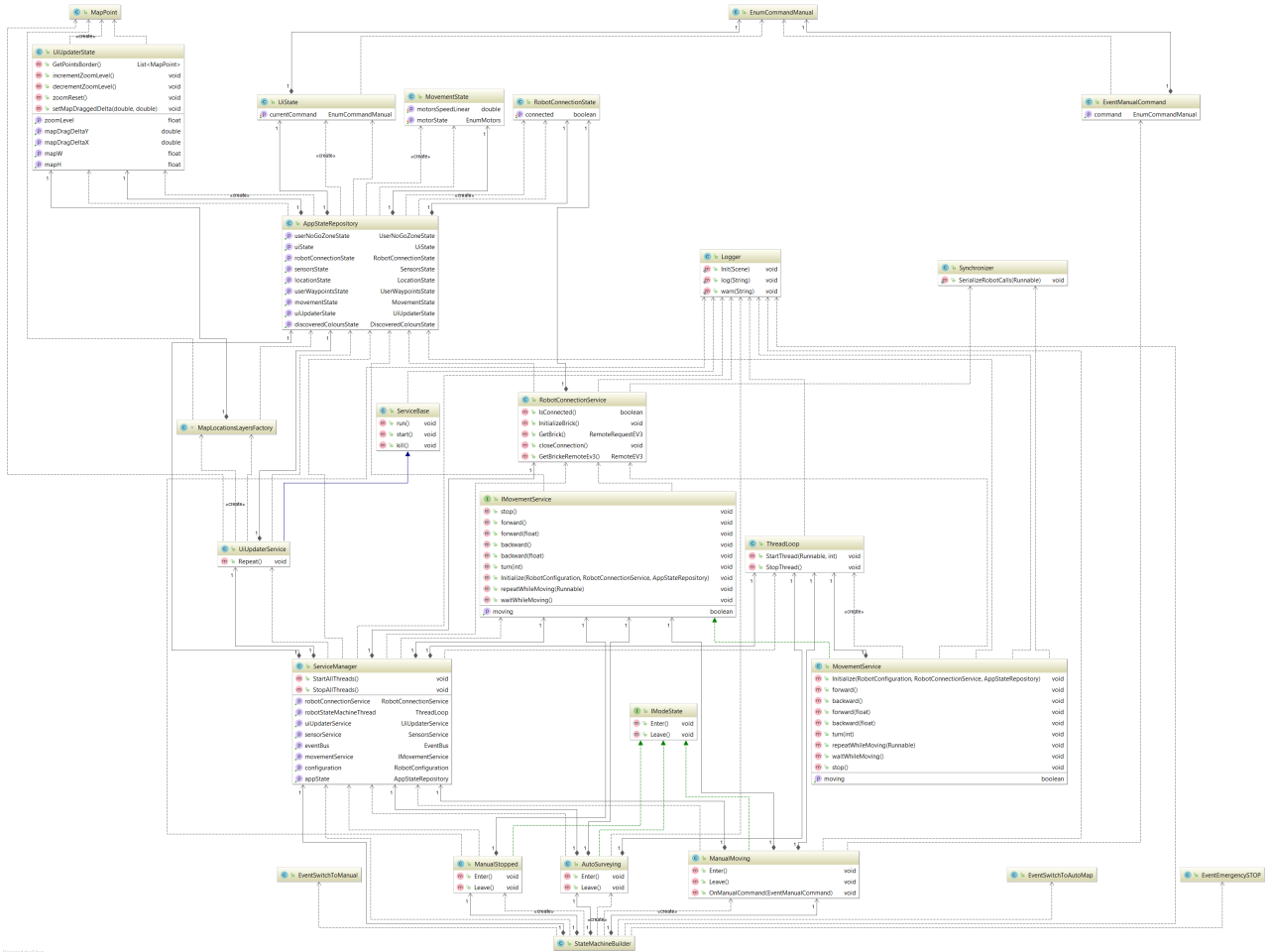


Figure 7: Class diagram of the whole system

## 5.2 State Diagram

The picture below shows the state diagram of Lunar Rover project. There three main states:

- Auto survey state is the state when robot autonomous moving and survey the map.
- Manual control state is the state when robot will move under control of user.
- Idle state is when robot is stop moving.

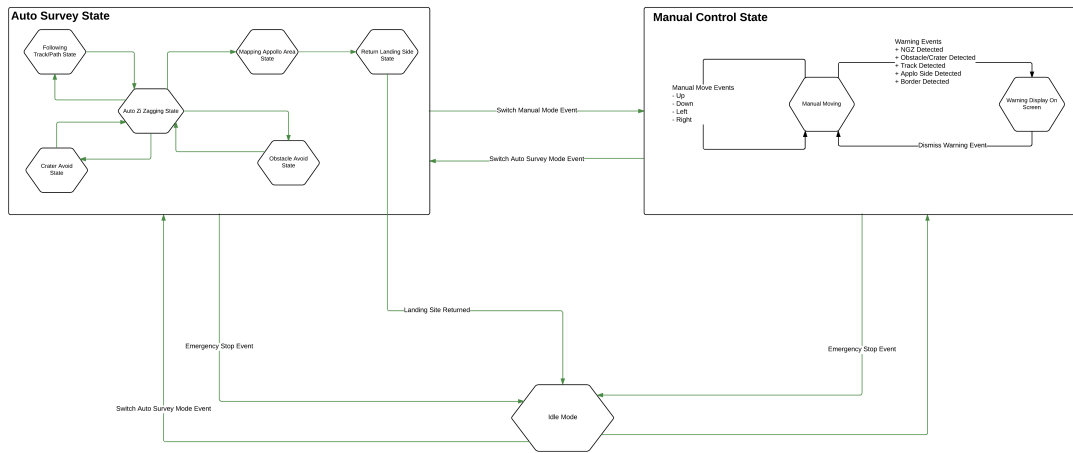


Figure 8: The State Diagram of the whole system

## 5.3 Interaction Diagram

The picture below shows the integration diagram of Lunar Rover system.

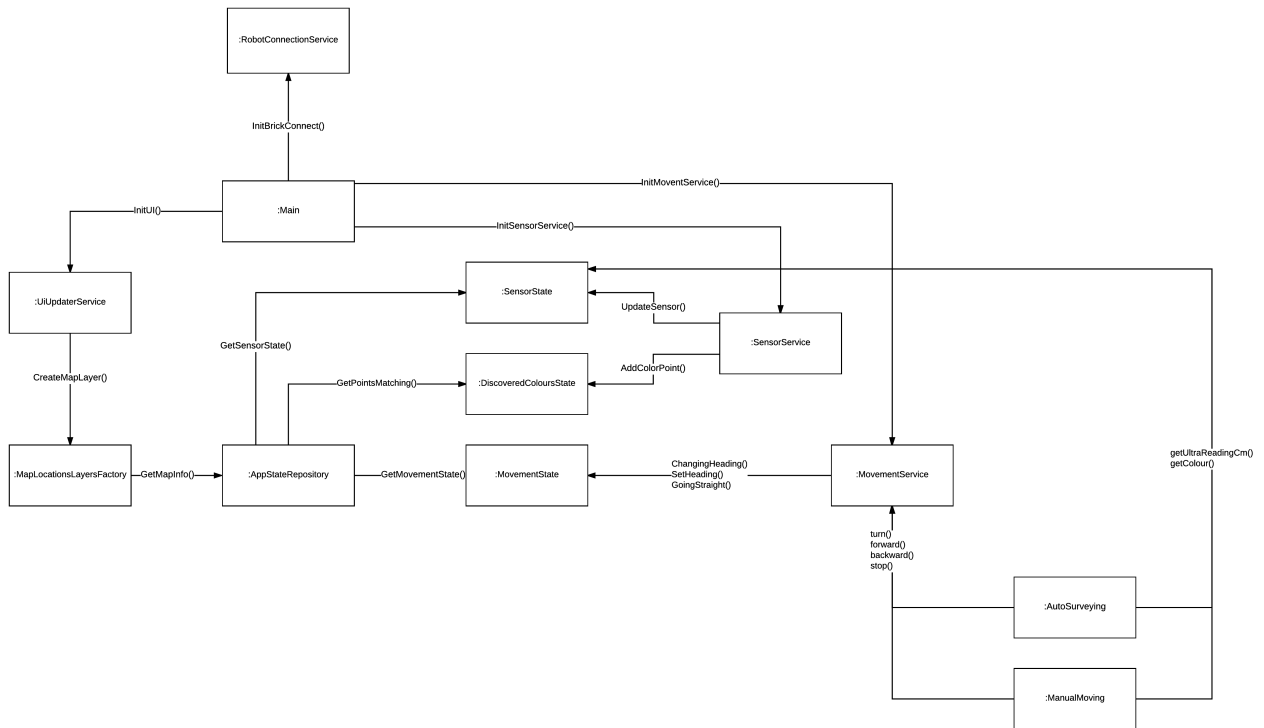


Figure 9: The interaction diagram of the whole system

## 6 Human Interface Design

### 6.1 Overview of the User Interface

According to Software Requirement Specification document(SRS), this embedded software system shall provide the graphic user interface for user to interact with the EV3 robot via graphic icons, file input and output. The below is the list of functionalities of GUI mentioned in SRS:

- The user is able to control the robot manually.
- The user is able to command the robot to move to specific point in the map under manual mode.
- The user is able to switch the current mode (Navigation and Manual modes) of the robot.
- The user is able to see the obstacles, line track displaced on the map.
- The user is able to trace the robot's current position.
- The user is able to see the detailed system log including software and hardware of the robot.
- The user is able to import and export XML file to define and retrieve the map information.

### 6.2 Detailed Design of the User Interface

The GUI is divided into four main parts and listed in Table 3. Each part of GUI consists of multiple panels.

Table 3: GUI Features

Name	Component	Related Features
Manual Control	C01, C05	UR13, UR14, FR008, FR009, FR010
Map panel	C06, C05	UR03, UR04, UR07, UR12, UR15, FR006, FR012
System status and log panels	C02, C05	FR015, EM001
Other functionalities	C05	UR05, UR06, FR001, FR002, FR003, FR004

#### 6.2.1 GUI's screen image

Fig.5 shows the screen image of GUI for end user perspective. The GUI supports mouse and keyboard devices. Each device is linked to GUI via action listeners established by GUI. When the GUI is created, there are multiply listeners created to listen the events from input devices. The detail actions will be discussed in section 6.2.2

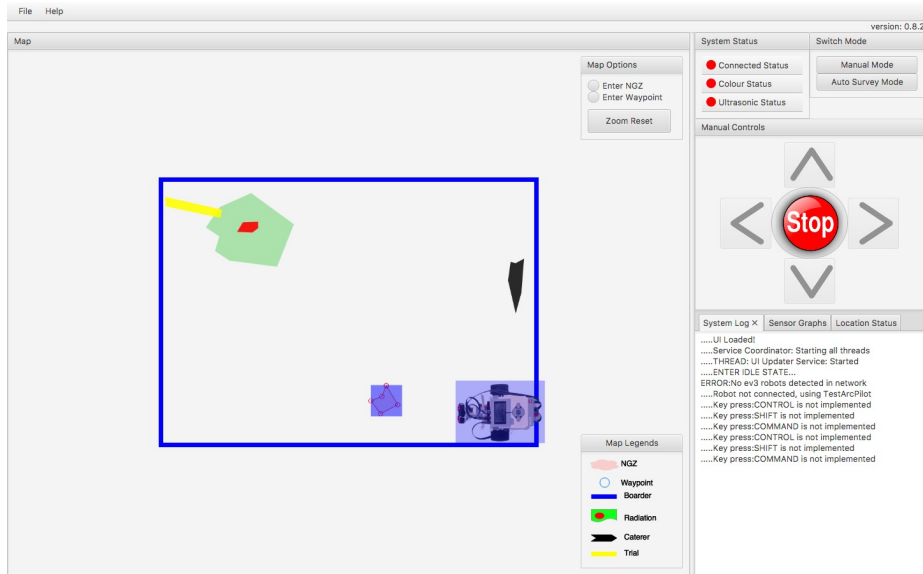


Figure 10: Graphic User Interface

### 6.2.2 Manual Control

#### Supporting input devices: Keyboard and Mouse

Manual Control has two listeners for listen the event and determine the actions. One listener is for the event coming from mouse device. The another is for the event coming from keyboard. Fig.6 is the manual control panel

#### Listener for listening mouse and keyboard event

The user can use their mouse to click the graphic icons showing on the Manual Control panel to control the robot. Also, the user can use keyboard to fire the event to GUI to control the robot instead. Each button and keyboard listeners have different method to control the robot. For example, if the user click the stop button, the method called `onClickStop()` will be called and stop the robot movement. The list of methods called by listeners is shown on Table 4.

### 6.2.3 Map panel

#### Supporting input devices: Mouse

The Map showing on Figure 6 is to display the area where the robot need to be explored. The user can select the NGZ area in the map and change size of the map. Also, the detected obstacles, NGZ area and track line will also be displayed in the map. The user can check the map legend to see what it looks in the map. Mouse listener will be created in the map to listen the event from the mouse. The list of methods called by listeners is shown on Table 5. Map also consists of different components. The below is the description of each component.

#### Map Area

Map area is shown in Figure 12a. The robot icon showing on the map area is represented as the current position of robot in the survey area. When the colour sensor of the robot is detecting the track/tail, the colour line will be drawn on

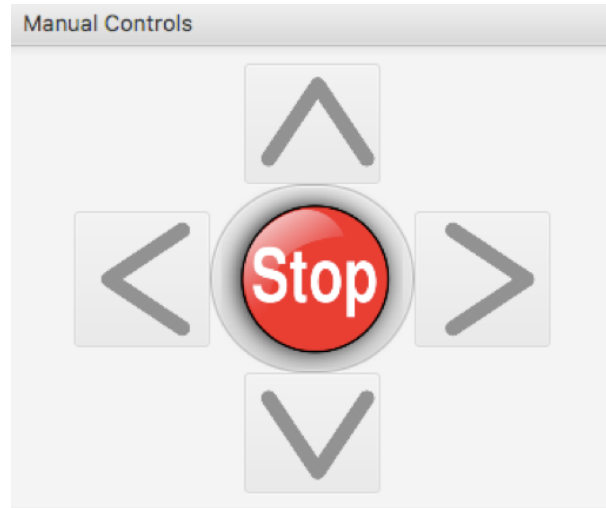







Figure 11: Manual control

Table 4: Method called for Manual Control

Method Name	Graphic Buttons	KeyBoard Keys	Action
<b>onClickForward</b>		W	Moving Forward
<b>onClickBackward</b>		S	Moving Backward
<b>onClickLeft</b>		A	Turn Left
<b>onClickRight</b>		D	Turn Right
<b>onClickStop</b>		Q	Stop the robot

the map. The map will be kept updating during the surveying.

### Map legends

The map legends are shown in Figure 12c. They provide a method of identifying items on the map area.

### Map Options

As shown in Figure 12b, if the user ticks the "Enter way point" box, then the user clicks on the map, the robot will go to that point. If the user ticks the "Enter NGZ" box, then the user can select the NGZ area on the map.

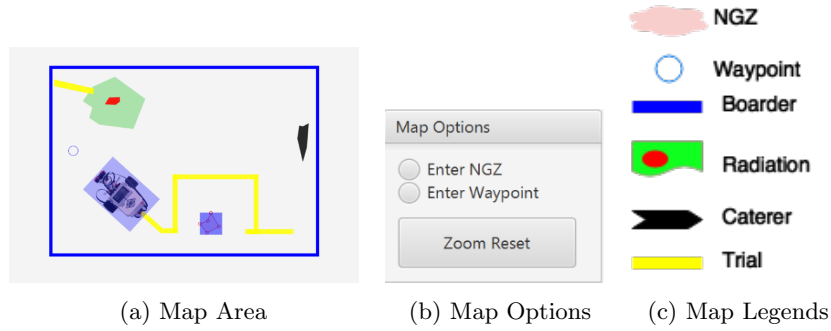


Figure 12: Map Panel Components

Table 5: Methods called by map listeners

Name	Mouse Action and Action
<b>onClickZoomReset</b>	Click on the reset button
<b>onClickMap</b>	Selecting the mode for map input, Enter NGZ or Enter waypoint
<b>onDragMap</b>	Determine the map action, the user do not need to concern this method

### Status and Logging Panels

The user can check the EV3 hardware status such as colour and ultrasonic sensors and switch the robot from manual mode to auto survey mode or menus through the status panel shown as Fig.13a. Also, the user can check the system log through the log console shown as Fig.13b.

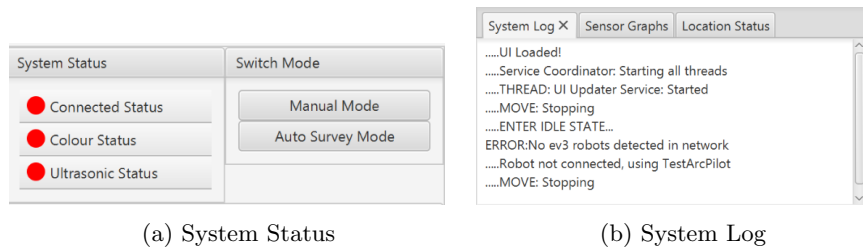


Figure 13: Status Panels

#### 6.2.4 Other Functionalities

The User can import and export the map XML file through the File button shown as Fig. 14. When the user import the map XML file, the map will show the obstacle, trail or track which are described in the XML file immediately. Also, if user has any operation issue, the use can click help button to find the manual for operating the software(shown as Fig. 15). The user can check the current version of the software through about button(show as Fig.16).

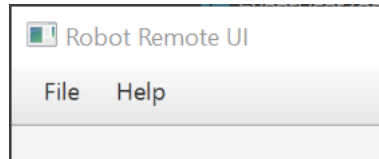


Figure 14: Other Menu Functions

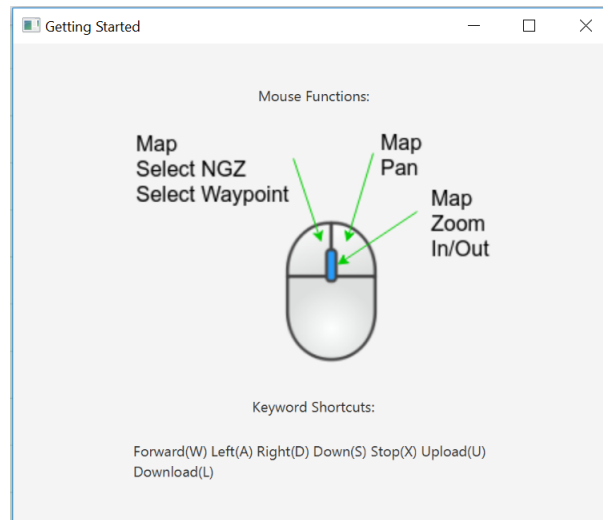


Figure 15: Help Popup Window





Figure 16: About Popup Window

## **7 Resources Estimate**

### **7.1 Budget**

For this project, time is budget and following is the list of different activities along with the weekly time allocation to work on them.

- 1 hour : (20min client meeting and 40 min group meeting)
- 2 x 2 hours : group meetings
- 1 - 2 hours : individual work for documentations
- 2 hours : individual work for coding
- 2 hours : discussion and group feedback
- 1 hours : planning and management

### **7.2 Manpower**

Note : Each team member will play multiple roles

- 1 x project manager
- 1 x document manager
- 1 x development manager
- 1 x testing manager
- 5 x coders
- 6 x document developers
- 6 x testers

### **7.3 Software**

- IntelliJ IDEA
- GitHub
- LaTeX
- Slack
- OS - Microsoft widows 7 or higher, Mac OS X 10 and Ubumtu 16.10
- Java JRE-8

### **7.4 Hardware**

- 1 x Lego Mindstorms EV3 Education edition robot KIT
- 6 x laptops - with clock speed of 1.6 GHz, RAM -512mb, 512mb free storage space
- 1 x A1-size map for prototype

### **7.5 Facilities**

- 1 x lab for group and client meeting

## 8 Definitions, Acronyms and Abbreviations

### 8.1 Definitions

- IntelliJ IDE - Java(IDE) for developing computer software
- GitHub - Source code control system
- LaTeX - Document preparation system or document editor
- Slack - Communication tool

### 8.2 Acronyms

- RMS - Robot Mapping System
- OS - Operating System
- JRE - Java Runtime Environment
- IDE - Integrated development environment
- NGZ - No Go Zone
- RMS - Robot Mapping System
- SDD - Software Design Document
- SRS - Software Requirements Specification
- XML - Extensible Mark-up Language
- WDDM - Windows Display Driver Model

### 8.3 Abbreviations

- min - minute